
Unity®
Unity小游戏开发简介
2023

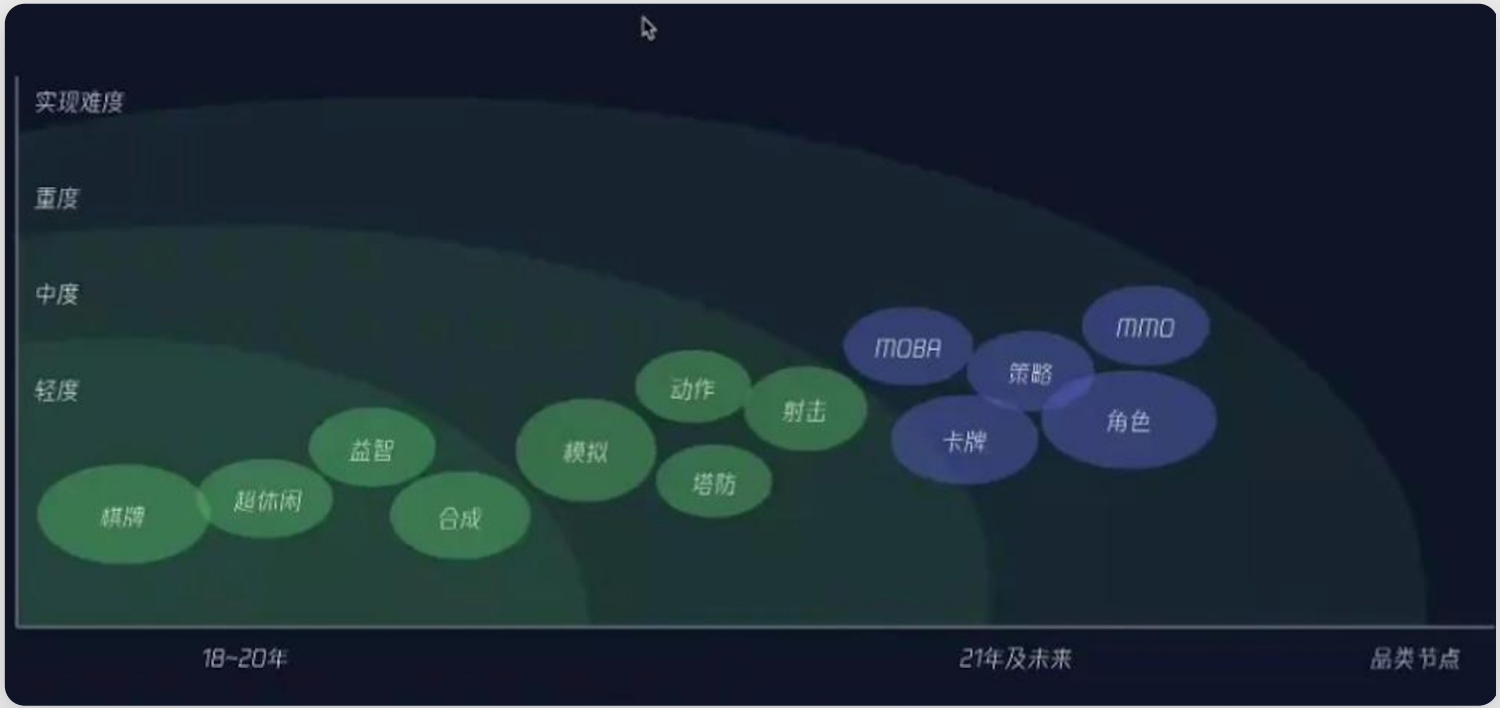
内容

- → 小游戏平台
- → 资源流式加载
- → Native Instant Game
- → WebGL
- → 近期进展
- → 未来工作

小游戏平台

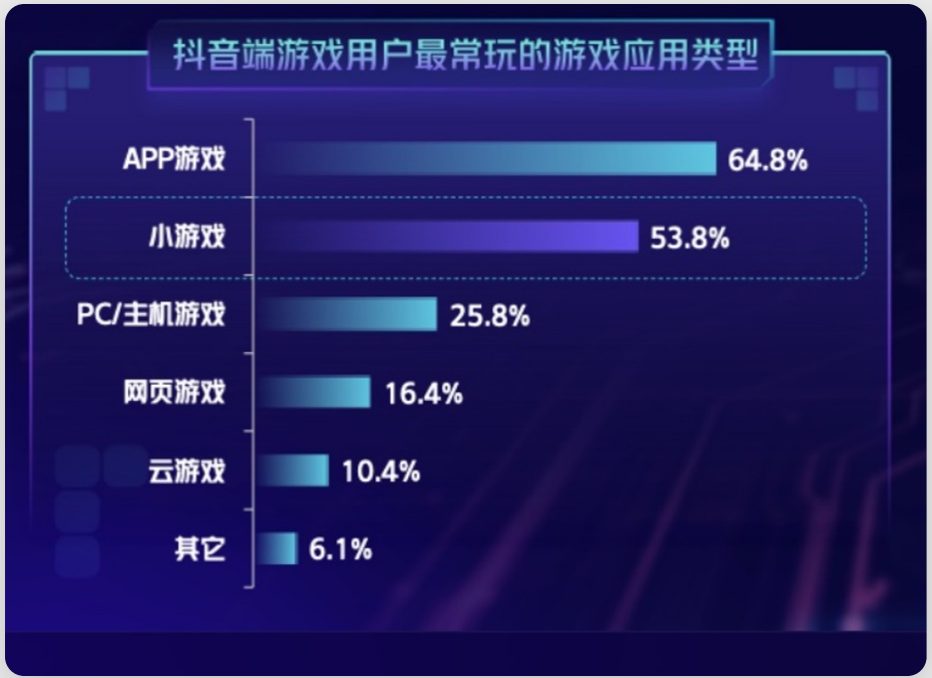
微信

微信



微信

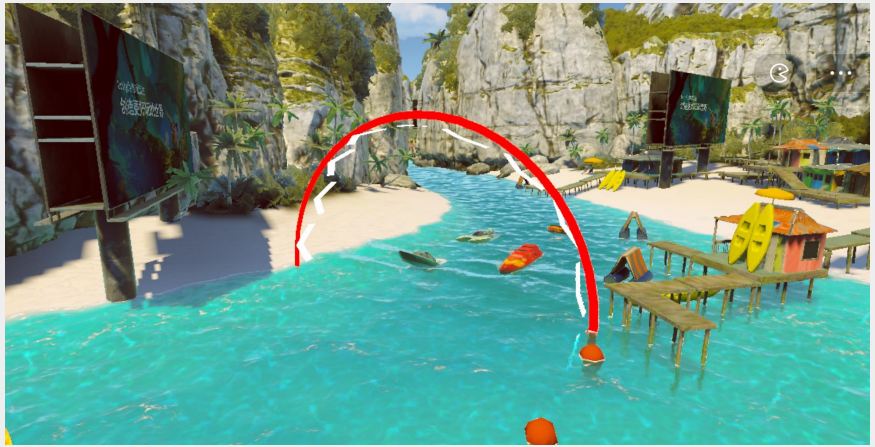
抖音



两种技术方案

WebGL

- 支持iOS、Android
- 各大平台均在使用

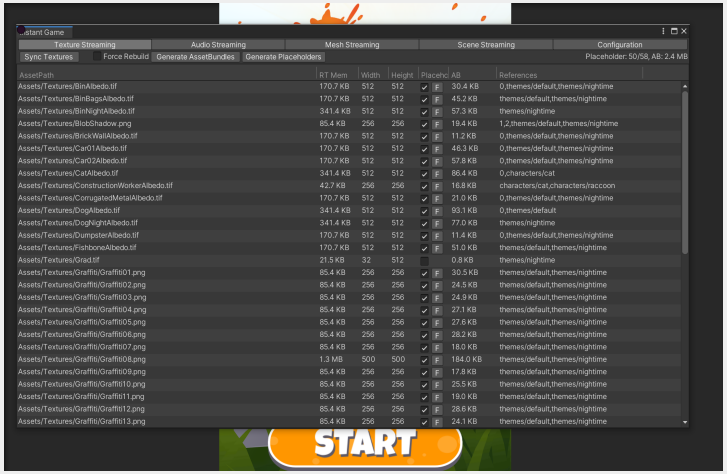


Native Instant Game

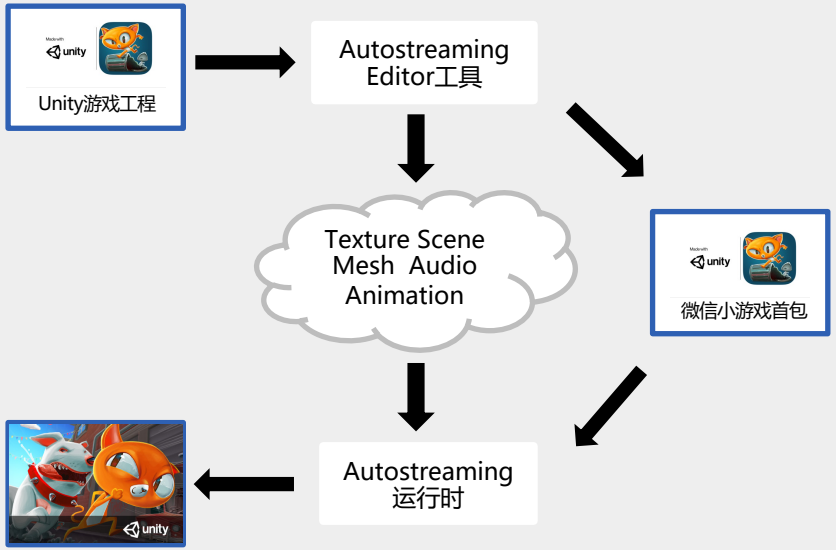
- 仅支持Android
- 原生App品质
- 已支持抖音、头条、快手
- 支付宝集成中

资源流式加载

1. 打包时自动分离重度资源



2. 运行时按需加载云端资源



优化效果

下载资源量对比

	未开启AutoStreaming	开启AutoStreaming
webgl.data压缩后	42.0MB	6.8MB
AssetBundles	50.6MB	32.5MB
启动耗时	40.0秒	7.88秒
内存占用	1265MB	1190MB

AutoStreaming云资源

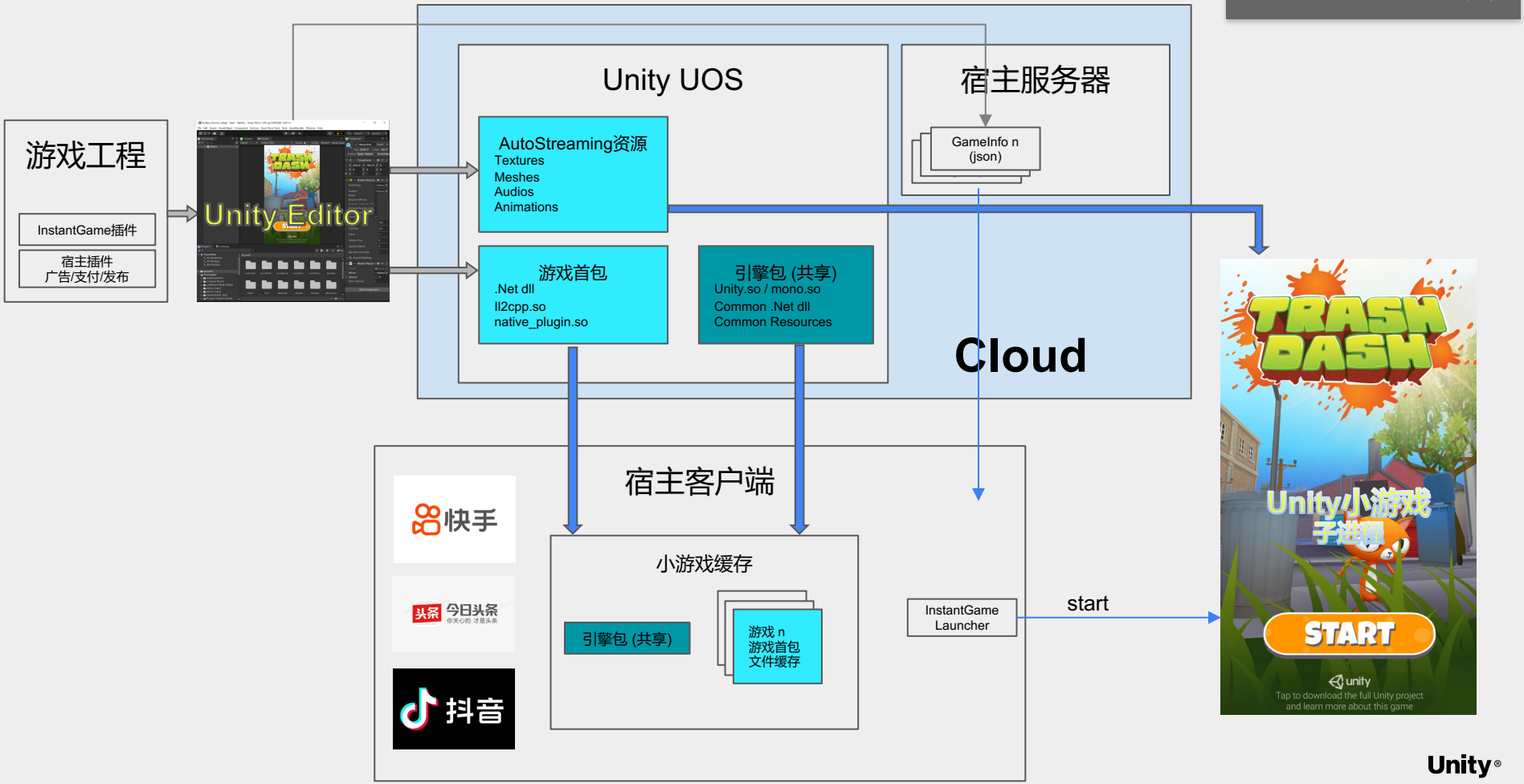
	个数	总大小
Texture	131	21.0MB
Audio	239	30.6MB
Scene	6	8MB
Animation	10	0.1MB

Native Instant Game

Native Instant Game

- 与原生App性能、体验一致
  - 支持多线程
  - 支持Gles3、Vulkan
- 兼容各种原生app能用的插件
- 可以访问沙盒中的文件
- 独立子进程运行在沙盒中
- 包含安全方案，提审、发布、更新方案
- 适配成本低
  - 只需要做好资源流式加载
  - 无需额外的适配与优化

https://unity.cn/instantgame/doc/2019/2019.4.29f1c110



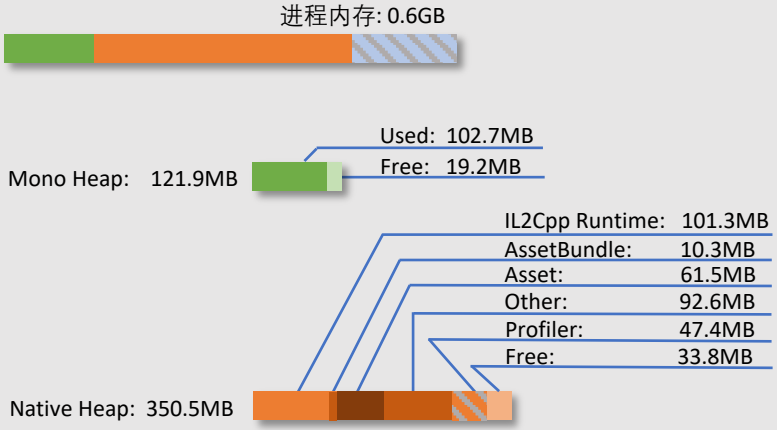

WebGL

平台特点

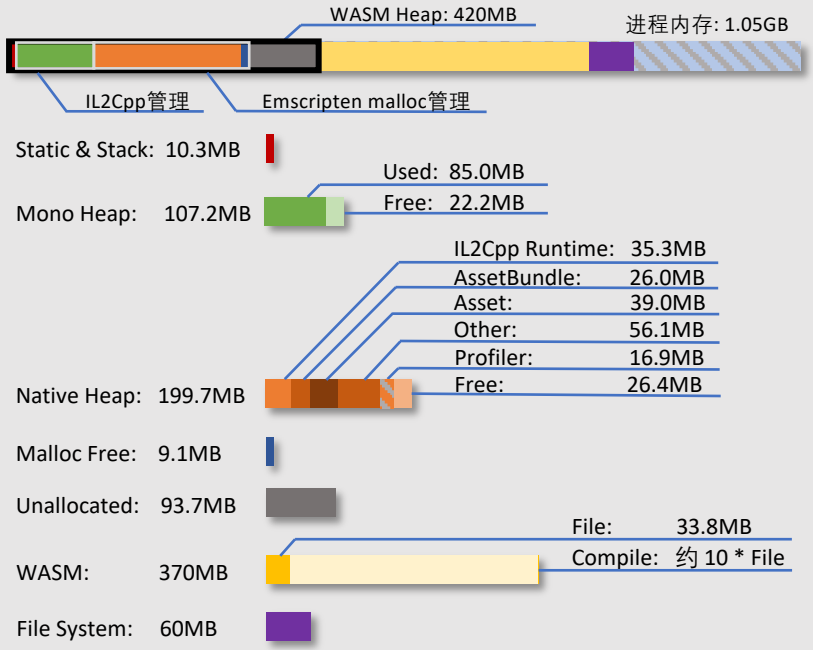
- → 内存
  - 在iOS上受限
- → CPU
  - Wasm, 单线程
- → GPU
  - WebGL1，WebGL2
- → 文件系统
  - 异步访问、费内存
- 发热

内存

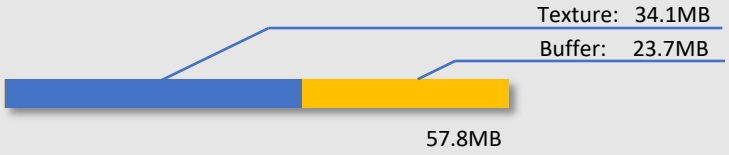
iOS 原生App

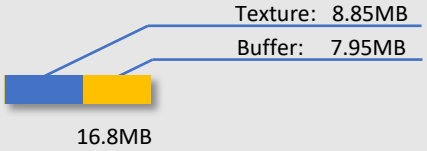


iOS WebGL (开启AutoStreaming)



显存

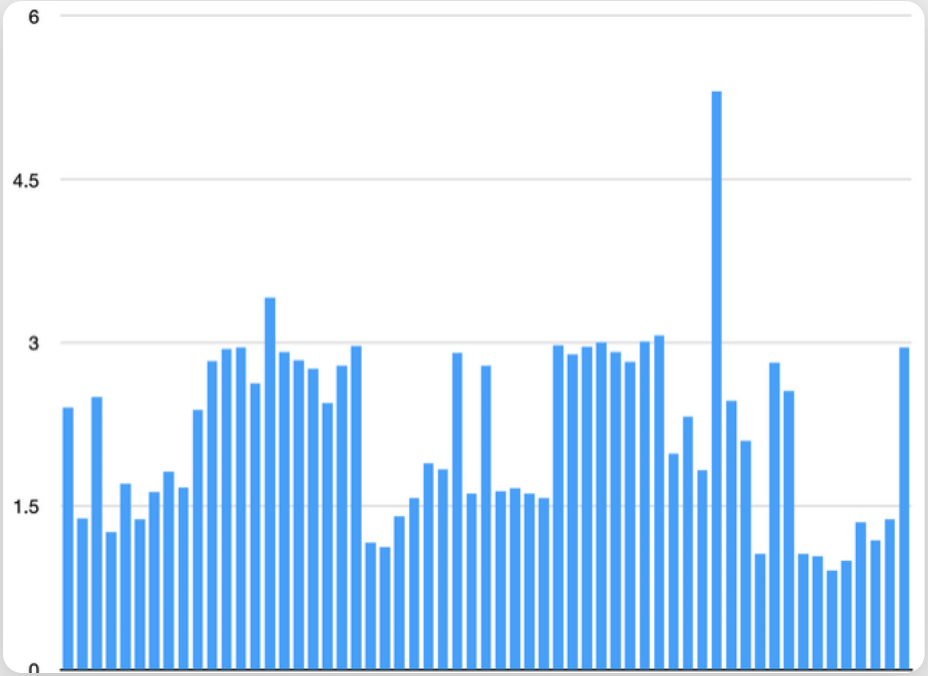




CPU

→ 3倍native耗时

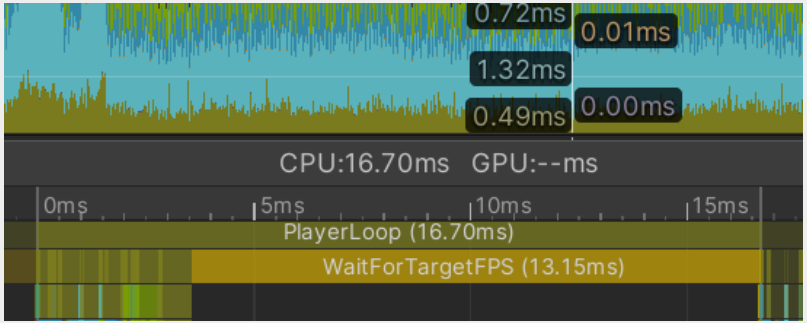
https://00f.net/2023/01/04/webassembly-benchmark-2023/

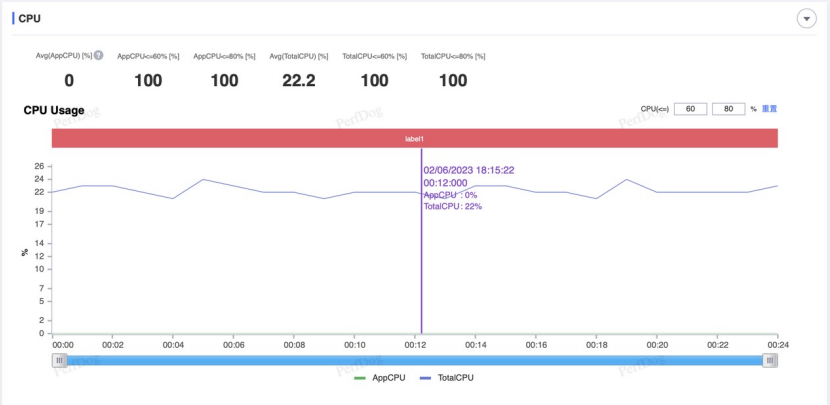


Wasm vs Native, 2023

CPU

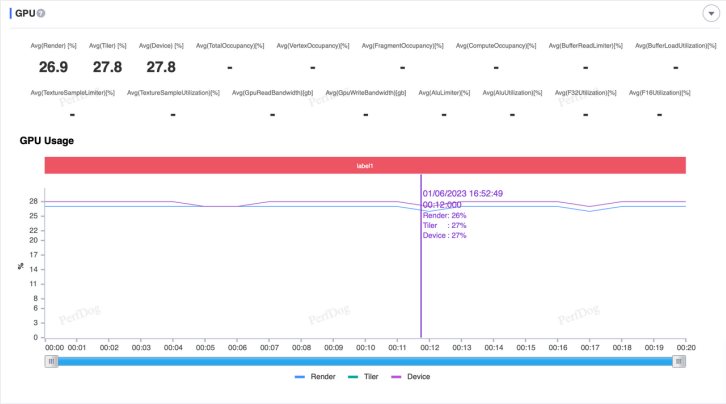
	iOS App	WebGL
单帧耗时(ms)	3.55ms	10.0ms
CPU 使用率	13.6%	22.2%

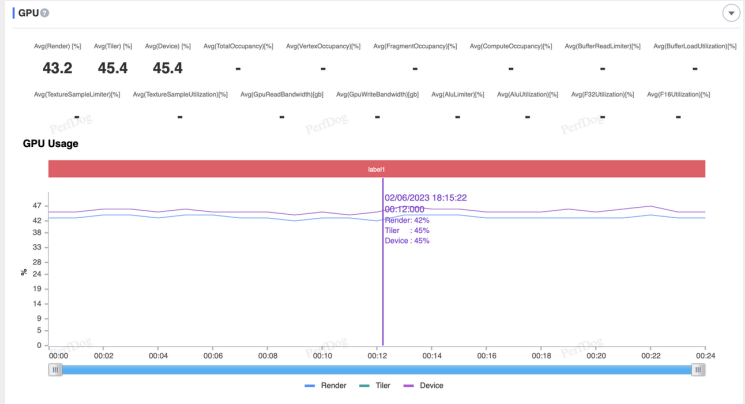




GPU

	iOS App (60fps)	WebGL (50fps)	空白网页 (30fps)
GPU 使用率	26.9%	43.2%	9.5%



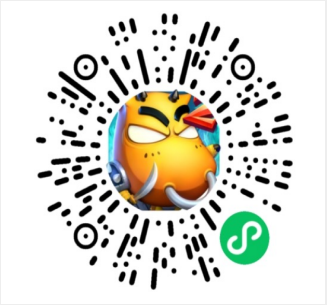


移植简介

- → 已经有大量的成功案例
- → 官方QQ群： 628540768
- → 微信WebGL适配教程

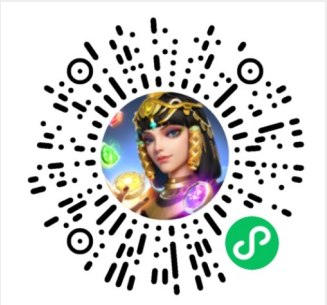
https://github.com/wechat-miniprogram/minigame-unity-webgl-transform

我叫MT2(回合战斗)

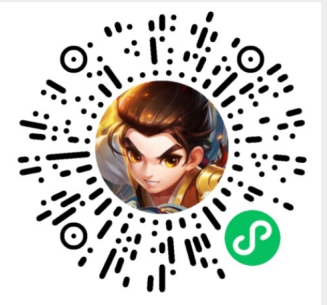


旅行串串(休闲)

谜题大陆(SLG)



热血神剑(MMO)



引擎改进

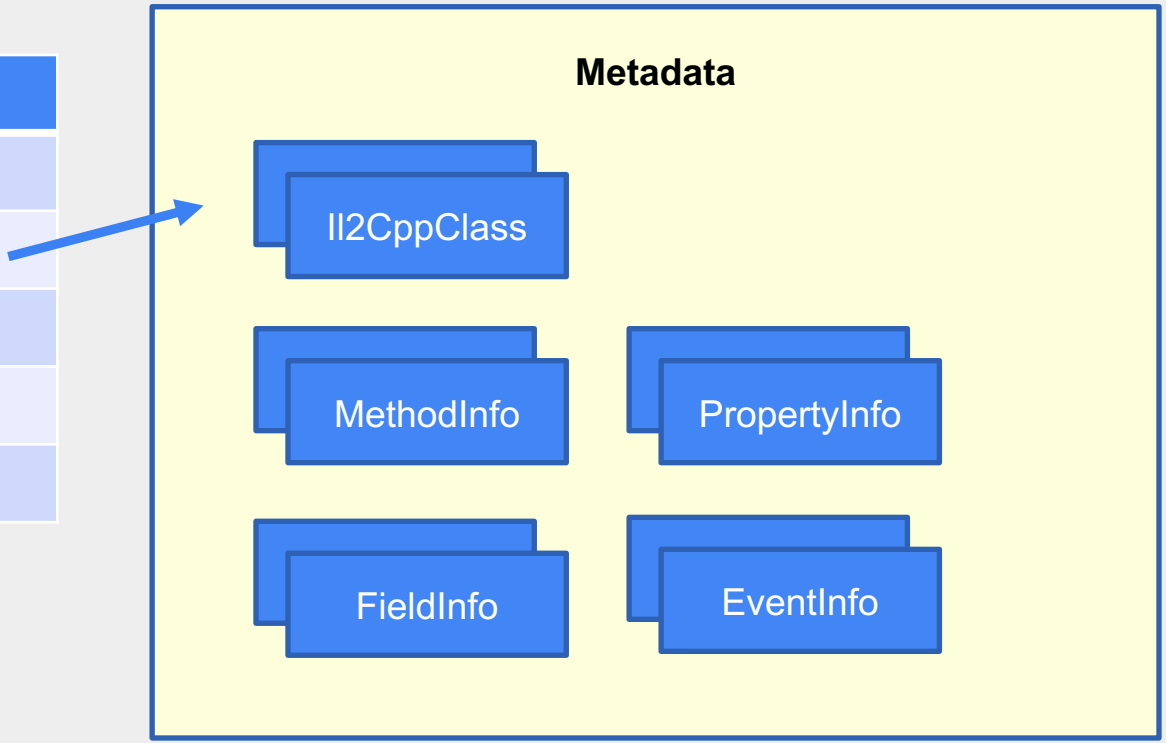
- 优化内存占用
- 优化绘制效率
- 引擎代码轻量化
- 加快启动速度

优化内存占用

- → 优化IL2CPP运行时内存占用 (64MB -> 33MB)
- → 优化DynamicVBO pool的复用机制，减少粒子系统等显存占有 (59MB -> 38MB)
- → 后面提到的代码轻量化、资源裁剪也会帮助减小运行时内存

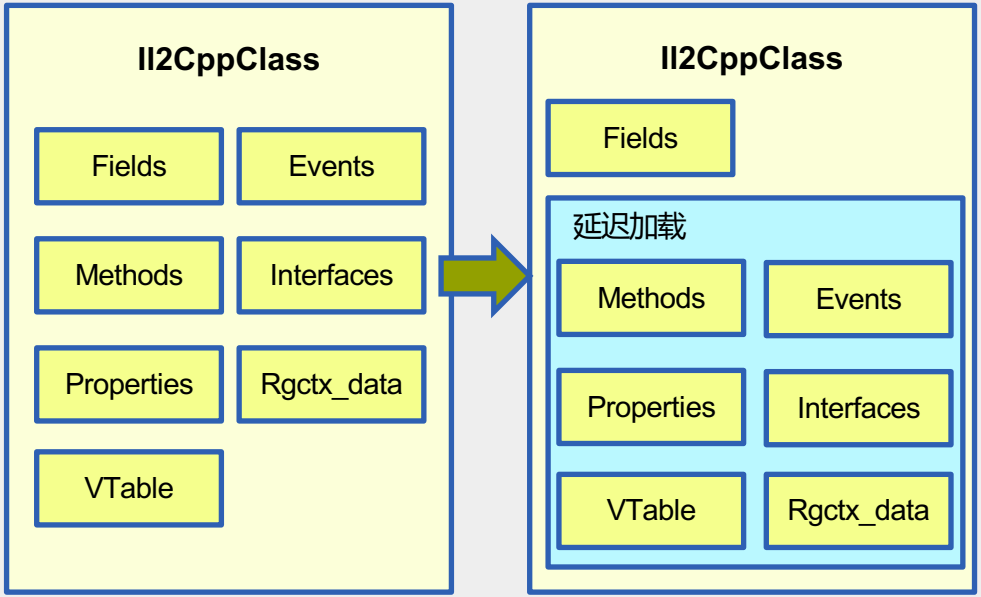
优化IL2CPP运行时内存占用

	案例1(MB)
总和	63.89
Metadata	37.03
global-metadata.dat	14.00
HashTable	7.96
Others	4.90



优化IL2CPP运行时内存占用

	优化前(MB)	优化后(MB)
Metadata总和	37.03	13.22
Class	9.63	3.43
Method	16.9	4.29
Property	0.85	0.001
GenericClass	0.48	0.10
GenericMethod	2.26	0.93



优化IL2CPP运行时内存占用

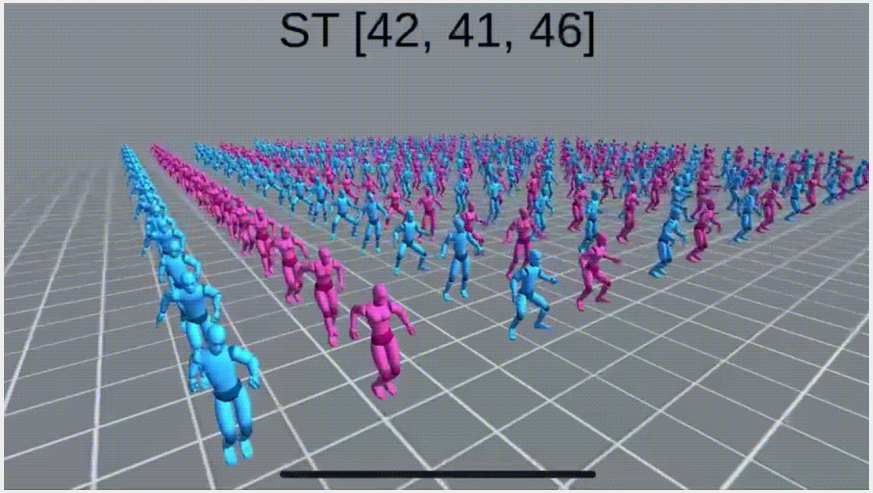
	案例1 优化前 (MB)	案例1 优化后 (MB)	案例2 优化前 (MB)	案例2 优化后 (MB)
总和	63.89	33.11	11.07	6.56
Metadata	37.03	13.22	5.02	1.99
global-metadata.dat	14.00	14.00	3.04	3.04
HashTable	7.96	1.90	2.18	0.72
Others	4.90	4.87	0.83	0.82

优化绘制效率

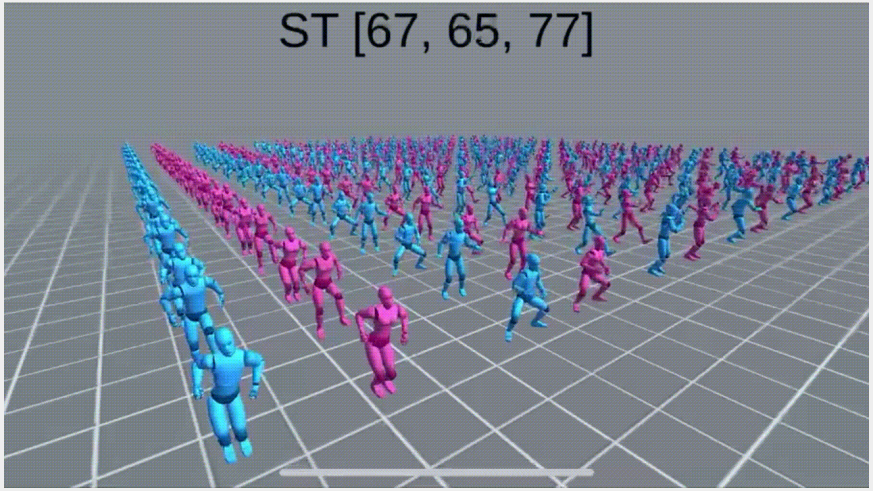
- → 在WebGL2上支持GPU skinning (15fps -> 24fps)
- → 优化Shader Compiler，将non-const global变量移到main函数中 (23fps -> 55fps)
- → 修改Immediate Const Buffer转换过程，声明成const并直接附初始值 (32fps -> 37fps)
- → 提供可配置的max visible lights值，32可降为16 (11fps -> 28fps)
- → iOS实现相关
  - 优化urpbatch在iOS上的行为，避免使用过多uniform变量，优化性能。
  - 在iOS14.x-15.4的WebGL上，同一个Canvas不共享IB和VB，改善UI渲染性能。

GPU Skinning

测试模型：4000顶点，53骨骼，每个顶点由4根骨骼控制
测试场景：320个模型，一共有8种动画
测试机型：iphone12+chrome



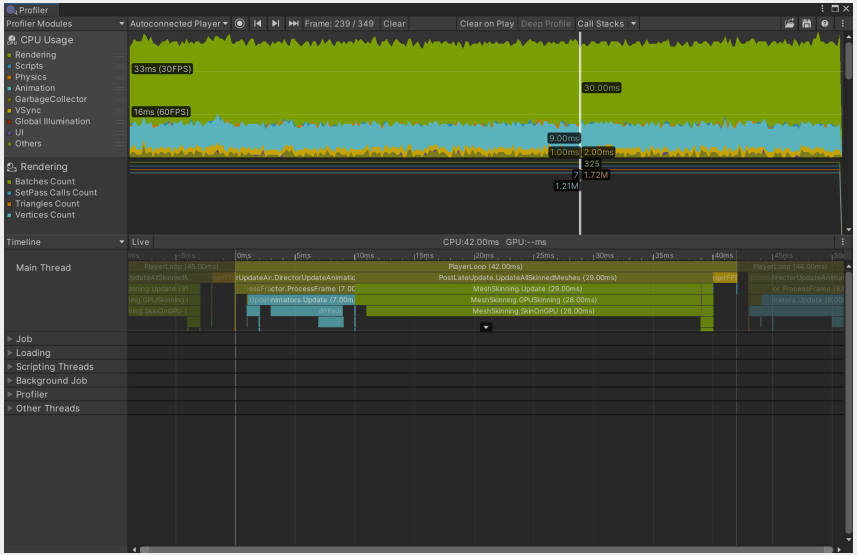
开启TF Skinning



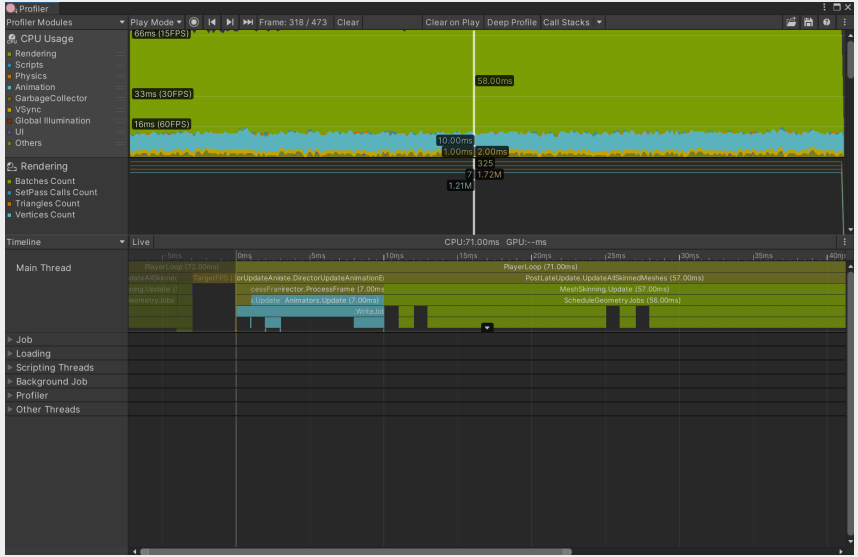
未开启TF Skinning

GPU Skinning

测试模型：4000顶点，53骨骼，每个顶点由4根骨骼控制
测试场景：320个模型，一共有8种动画
测试机型：iphone12+chrome



开启TF Skinning

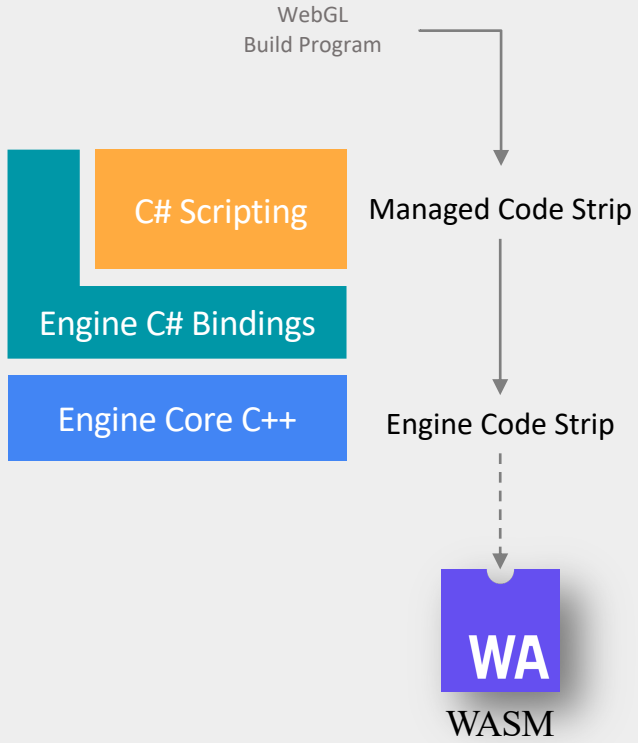


未开启TF Skinning

引擎代码轻量化

→ 当前方法

- Managed Code Strip
- Engine Code Strip
- 它们是通过静态分析依赖的方式做的strip
- 以函数为粒度

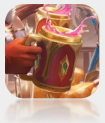


引擎代码轻量化

Wasm.br →

WASM →

Brotli



案例1

案例2

	案例1		案例2	
压缩大小 / br.Size	7.3 MB		7.0 MB	
解压后大小 / Size	35.1 MB		32.6 MB	
解析指令数 / Instructions	12.8 M		12.0 M	
▪ IL2CPP-gen	7.7 M	~60 %	7.5 M	~60 %
▪ Engine C++		~40 %		~40 %
Physx	1.0 M	8 %	0.99 M	8 %
ParticleSystem	370 K	3 %	370 K	3 %
Sqlite	130 K	1 %	133 K	1 %
Mecanim	68 K	.5%	67 K	.5%
…				
Non-Modular	1.3 M	10 %	1.5 M	13 %

引擎代码轻量化

→ 进一步轻量化

- 将C++模板参数改为函数参数，减少生成的代码量
- 通过宏隔离无法执行到的逻辑，避免引入不必要的依赖

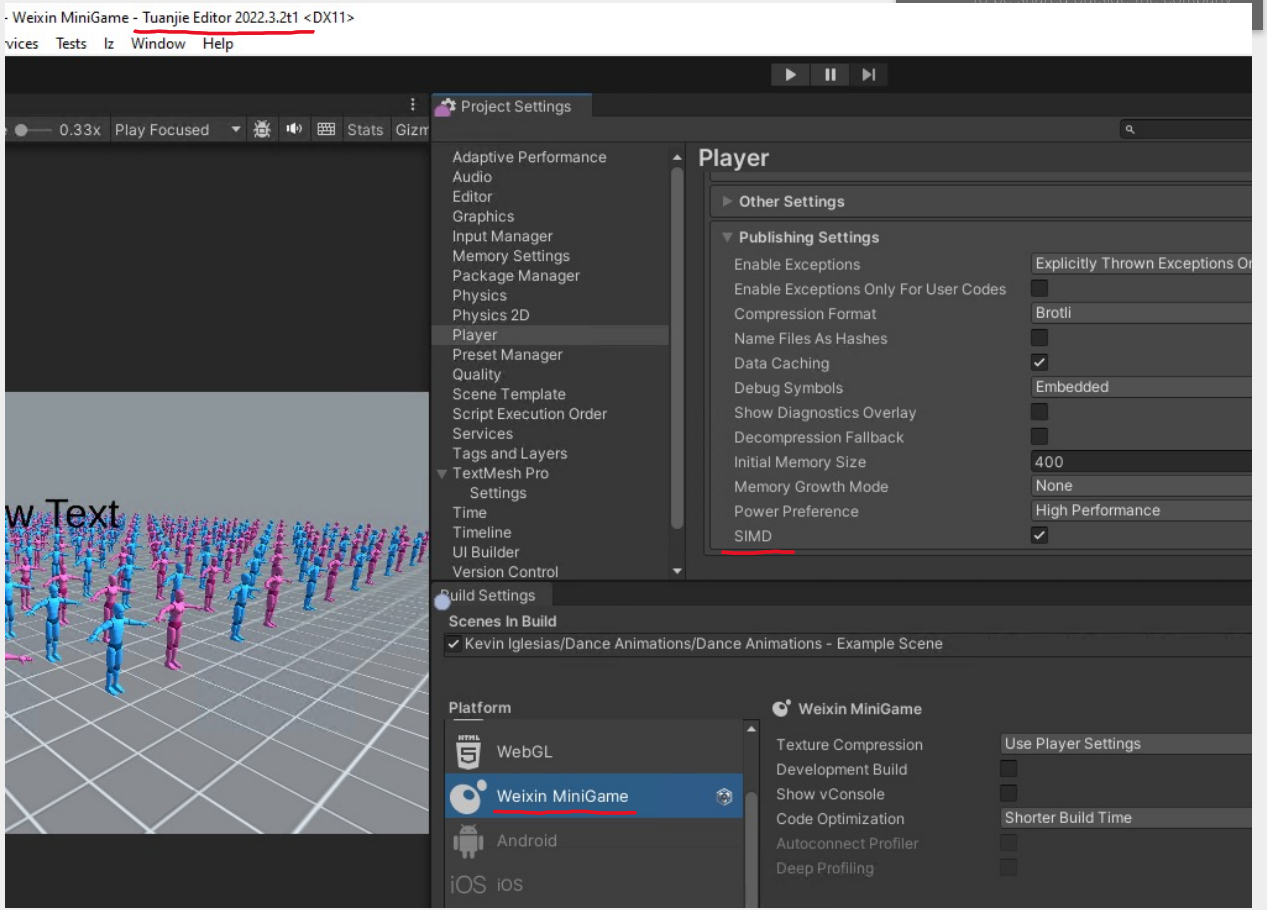
加快启动速度

- → 与平台合作，使用宿主提供的中文字体
- → 动态裁减unity default resource
- → 尝试wasm snapshot方案

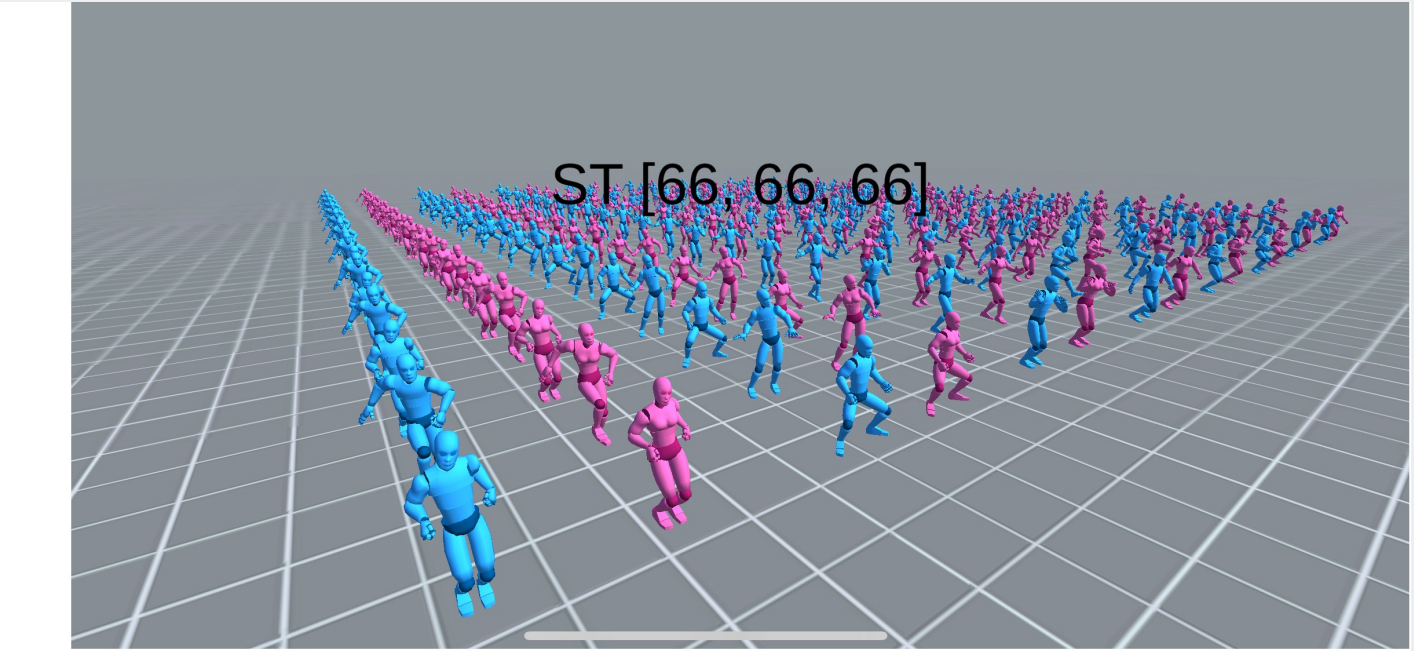
近期进展

SIMD

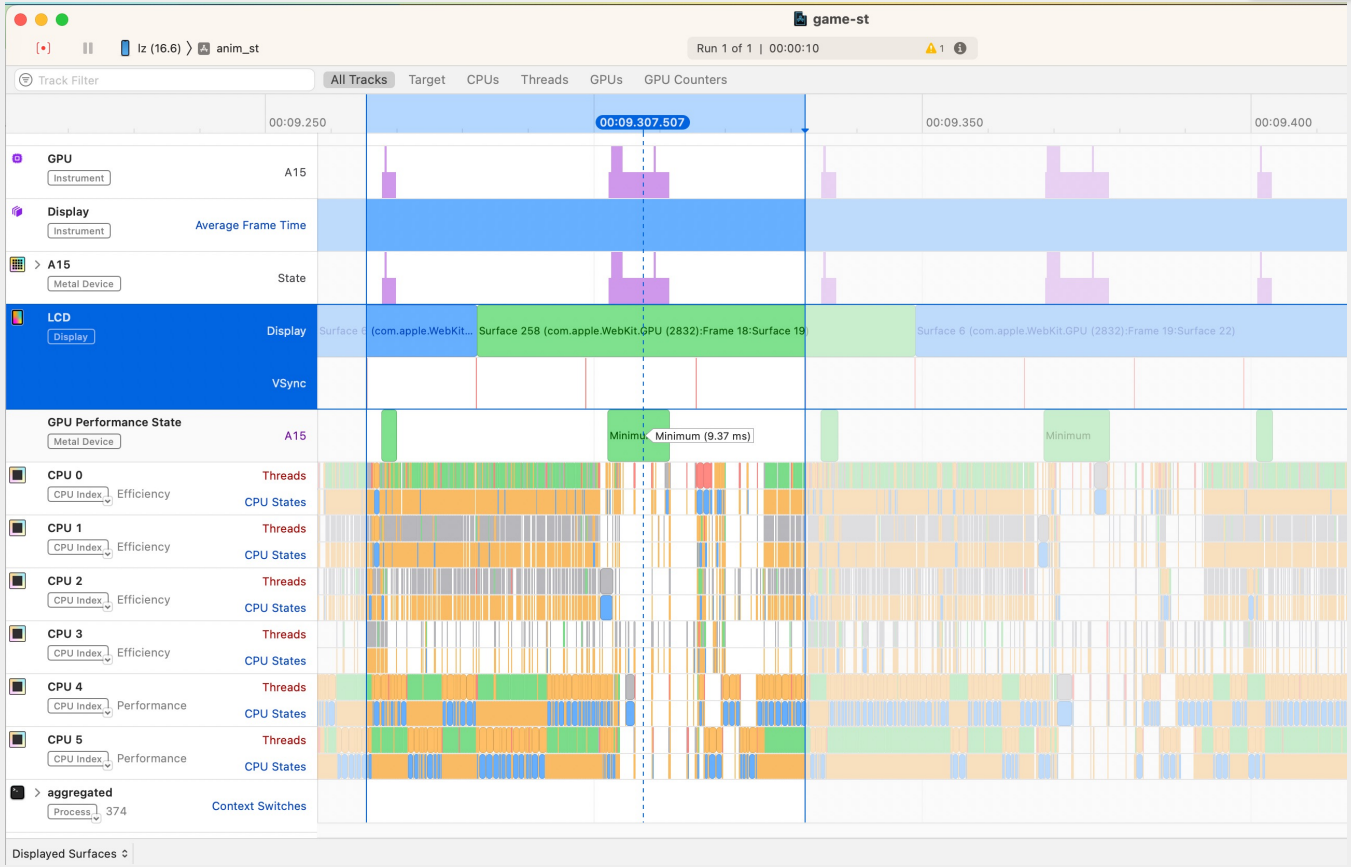
- → 增加SIMD支持
  - Fixed width 128位
  - 适配Mesh Skinning
  - iOS Safari 16.4
  - Android Chrome 91
- → 未来会适配math库
  - Animation、particle system都将收益



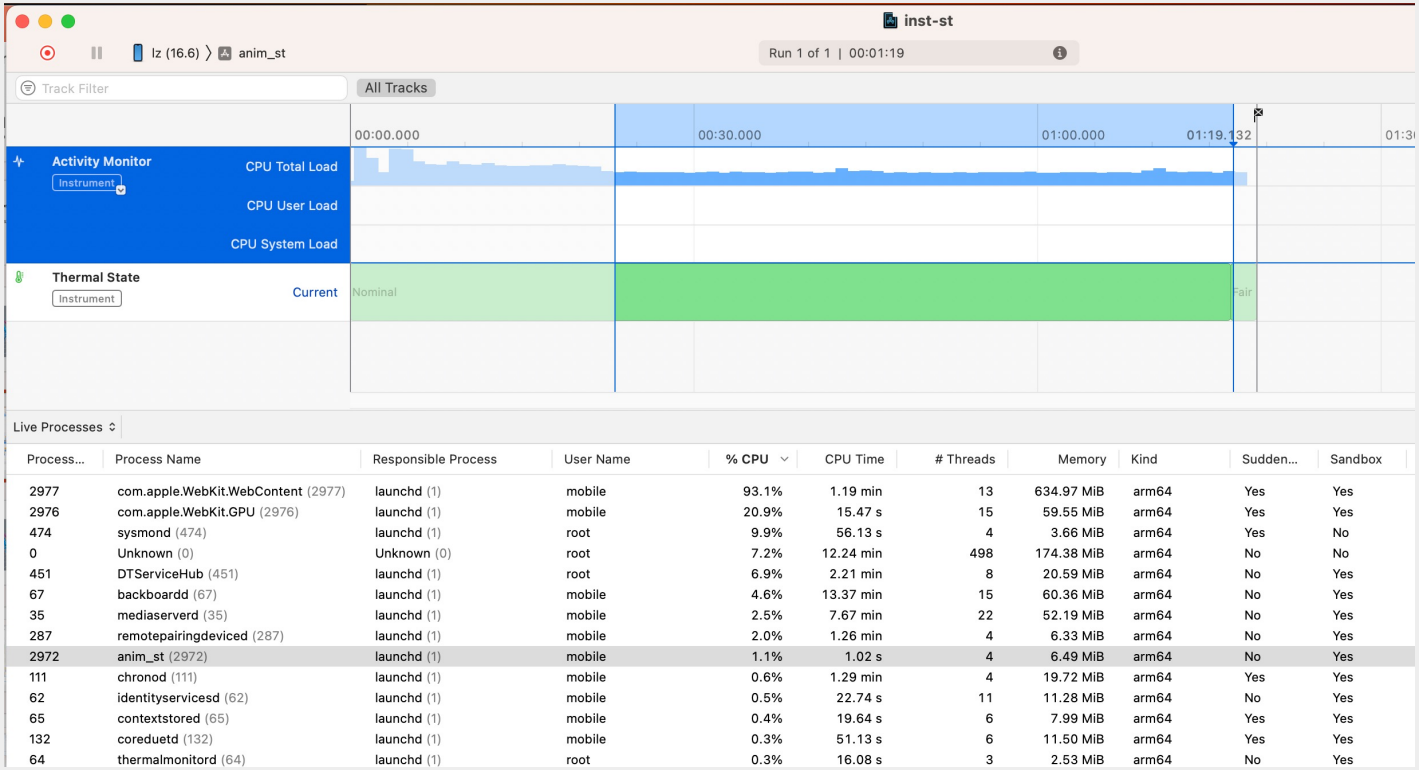
St



St

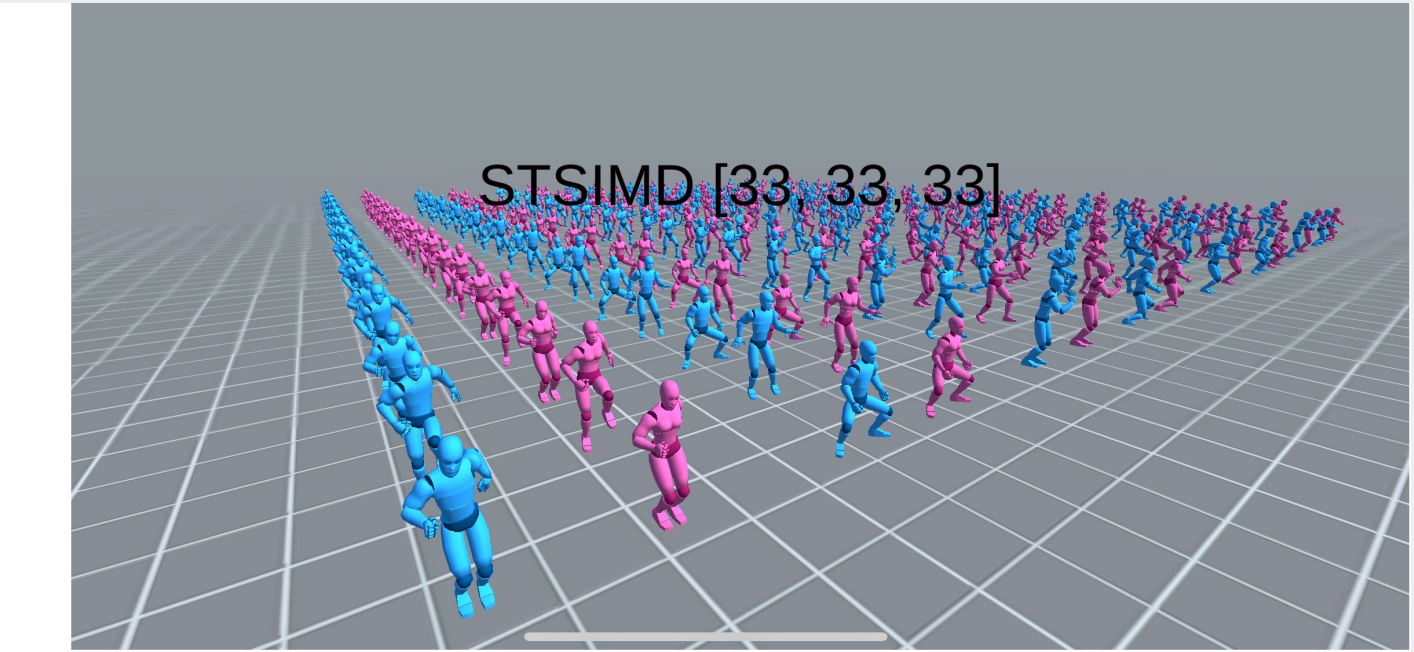


St

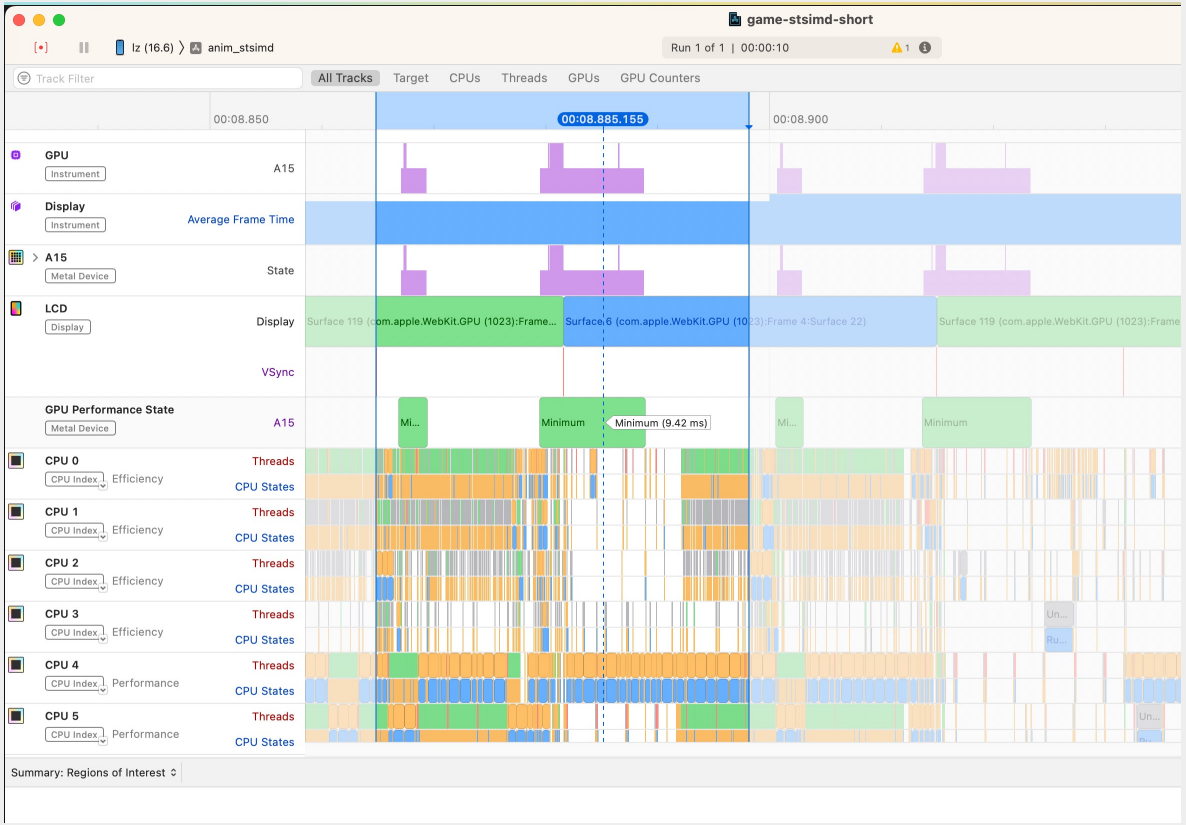


Process...	Process Name	Responsible Process	User Name	% CPU	CPU Time	# Threads	Memory	Kind	Sudden...	Sandbox
2977	com.apple.WebKit.WebContent (2977)	launchd (1)	mobile	93.1%	1.19 min	13	634.97 MiB	arm64	Yes	Yes
2976	com.apple.WebKit.GPU (2976)	launchd (1)	mobile	20.9%	15.47 s	15	59.55 MiB	arm64	Yes	Yes
474	sysmond (474)	launchd (1)	root	9.9%	56.13 s	4	3.66 MiB	arm64	Yes	No
0	Unknown (0)	Unknown (0)	root	7.2%	12.24 min	498	174.38 MiB	arm64	No	No
451	DTServiceHub (451)	launchd (1)	root	6.9%	2.21 min	8	20.59 MiB	arm64	No	Yes
67	backboardd (67)	launchd (1)	mobile	4.6%	13.37 min	15	60.36 MiB	arm64	No	Yes
35	mediaserverd (35)	launchd (1)	mobile	2.5%	7.67 min	22	52.19 MiB	arm64	No	Yes
287	remotepairingdeviced (287)	launchd (1)	mobile	2.0%	1.26 min	4	6.33 MiB	arm64	No	Yes
2972	anim_st (2972)	launchd (1)	mobile	1.1%	1.02 s	4	6.49 MiB	arm64	No	Yes
111	chronod (111)	launchd (1)	mobile	0.6%	1.29 min	4	19.72 MiB	arm64	Yes	Yes
62	identityservicesd (62)	launchd (1)	mobile	0.5%	22.74 s	11	11.28 MiB	arm64	No	Yes
65	contextstored (65)	launchd (1)	mobile	0.4%	19.64 s	6	7.99 MiB	arm64	Yes	Yes
132	coreduetd (132)	launchd (1)	mobile	0.3%	51.13 s	6	11.50 MiB	arm64	Yes	Yes
64	thermalmonitord (64)	launchd (1)	root	0.3%	16.08 s	3	2.53 MiB	arm64	No	Yes

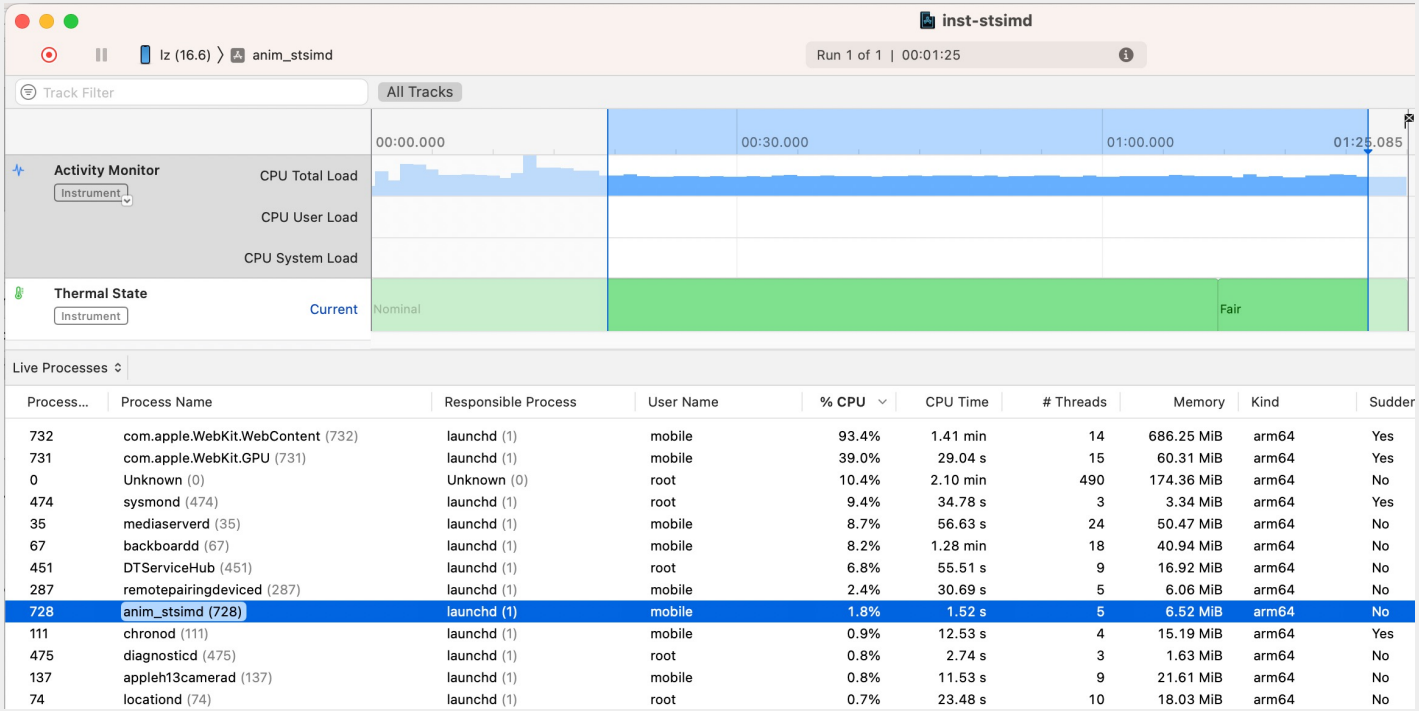
StSimd



StSimd

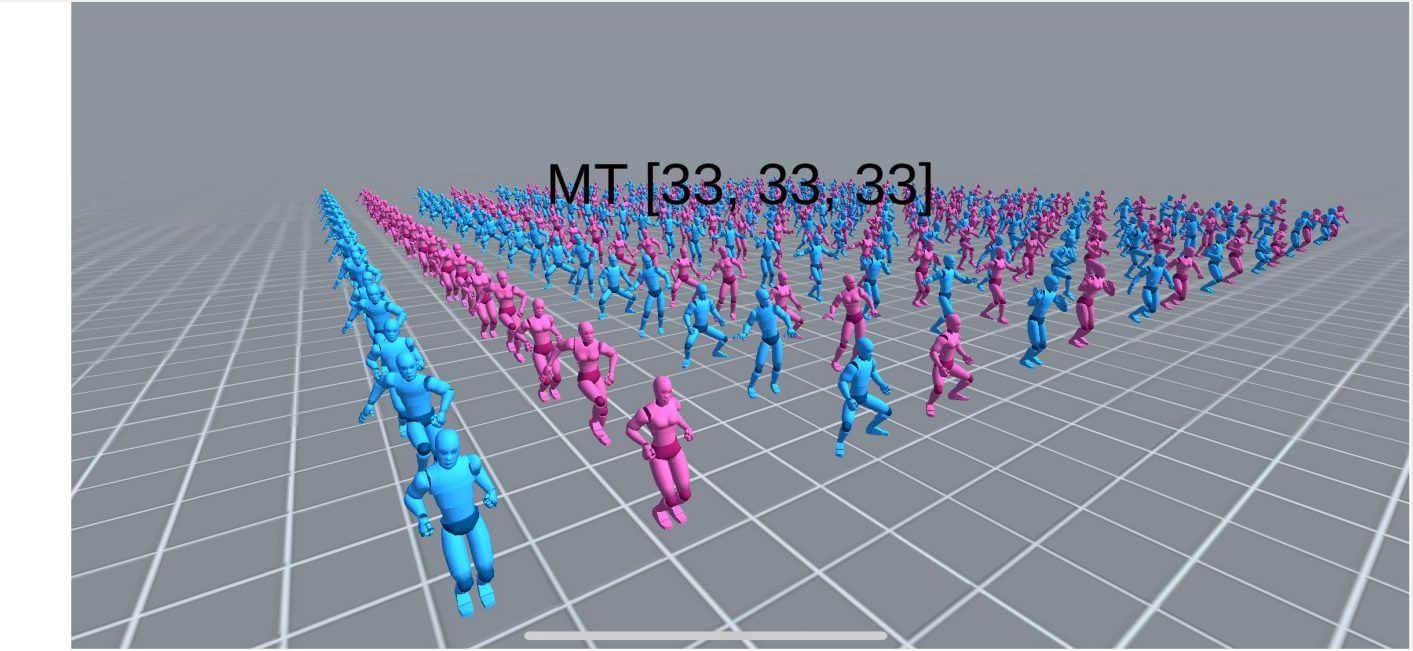


StSimd

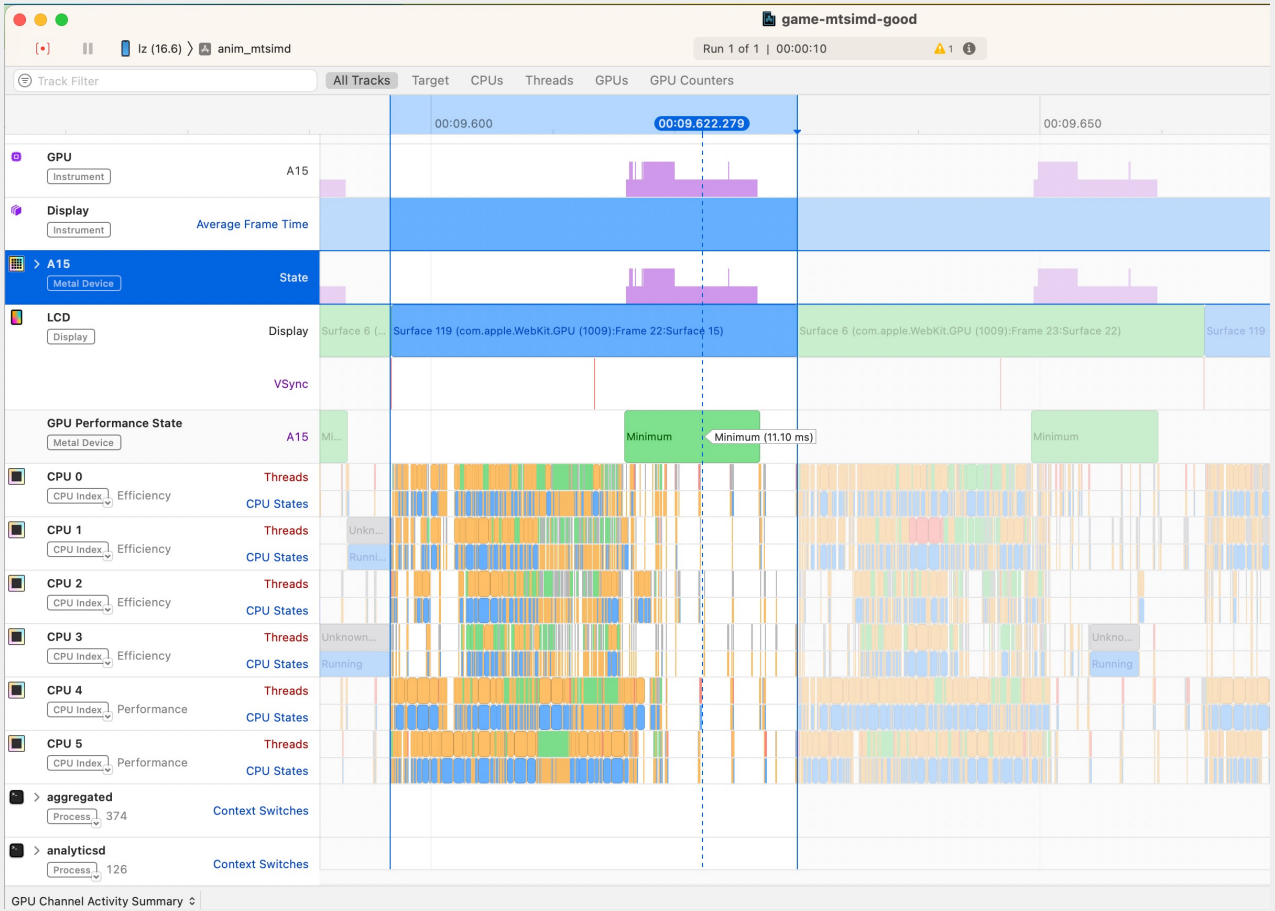


Process...	Process Name	Responsible Process	User Name	% CPU	CPU Time	# Threads	Memory	Kind	Sudder
732	com.apple.WebKit.WebContent (732)	launchd (1)	mobile	93.4%	1.41 min	14	686.25 MiB	arm64	Yes
731	com.apple.WebKit.GPU (731)	launchd (1)	mobile	39.0%	29.04 s	15	60.31 MiB	arm64	Yes
0	Unknown (0)	Unknown (0)	root	10.4%	2.10 min	490	174.36 MiB	arm64	No
474	sysmond (474)	launchd (1)	root	9.4%	34.78 s	3	3.34 MiB	arm64	Yes
35	mediaserverd (35)	launchd (1)	mobile	8.7%	56.63 s	24	50.47 MiB	arm64	No
67	backboardd (67)	launchd (1)	mobile	8.2%	1.28 min	18	40.94 MiB	arm64	No
451	DTServiceHub (451)	launchd (1)	root	6.8%	55.51 s	9	16.92 MiB	arm64	No
287	remotepairingdeviced (287)	launchd (1)	mobile	2.4%	30.69 s	5	6.06 MiB	arm64	No
728	anim_stsimd (728)	launchd (1)	mobile	1.8%	1.52 s	5	6.52 MiB	arm64	No
111	chronod (111)	launchd (1)	mobile	0.9%	12.53 s	4	15.19 MiB	arm64	Yes
475	diagnosticd (475)	launchd (1)	root	0.8%	2.74 s	3	1.63 MiB	arm64	No
137	appleh13camerad (137)	launchd (1)	mobile	0.8%	11.53 s	9	21.61 MiB	arm64	No
74	locationd (74)	launchd (1)	root	0.7%	23.48 s	10	18.03 MiB	arm64	No

MtSimd



MtSimd

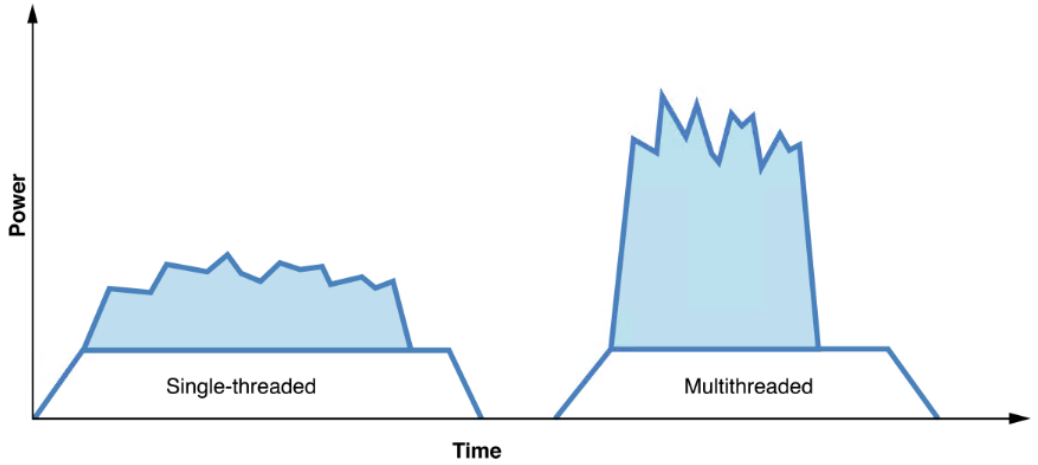


MtSimd

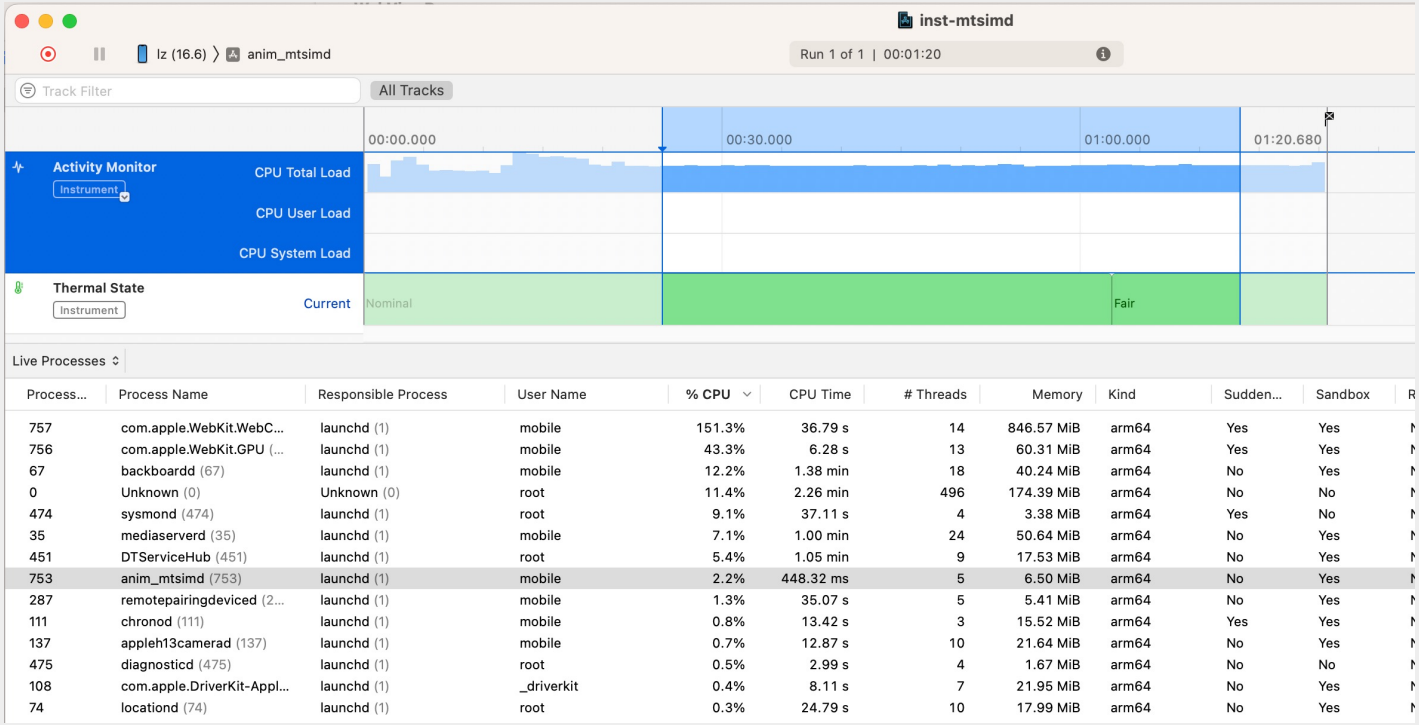
This strategy incurs a greater up-front dynamic cost—more work is done at a given time, requiring more power. In exchange, you get a dramatic reduction in fixed cost, which results in tremendous energy savings over time. You app draws more power, but it does so more efficiently and over less time. This lets the CPU get back to idle and other components to power down much more quickly.

As you develop your app, think holistically about its behavior, and try to reduce fixed costs wherever possible.

Figure 2-3 Use multithreading to trade power for energy

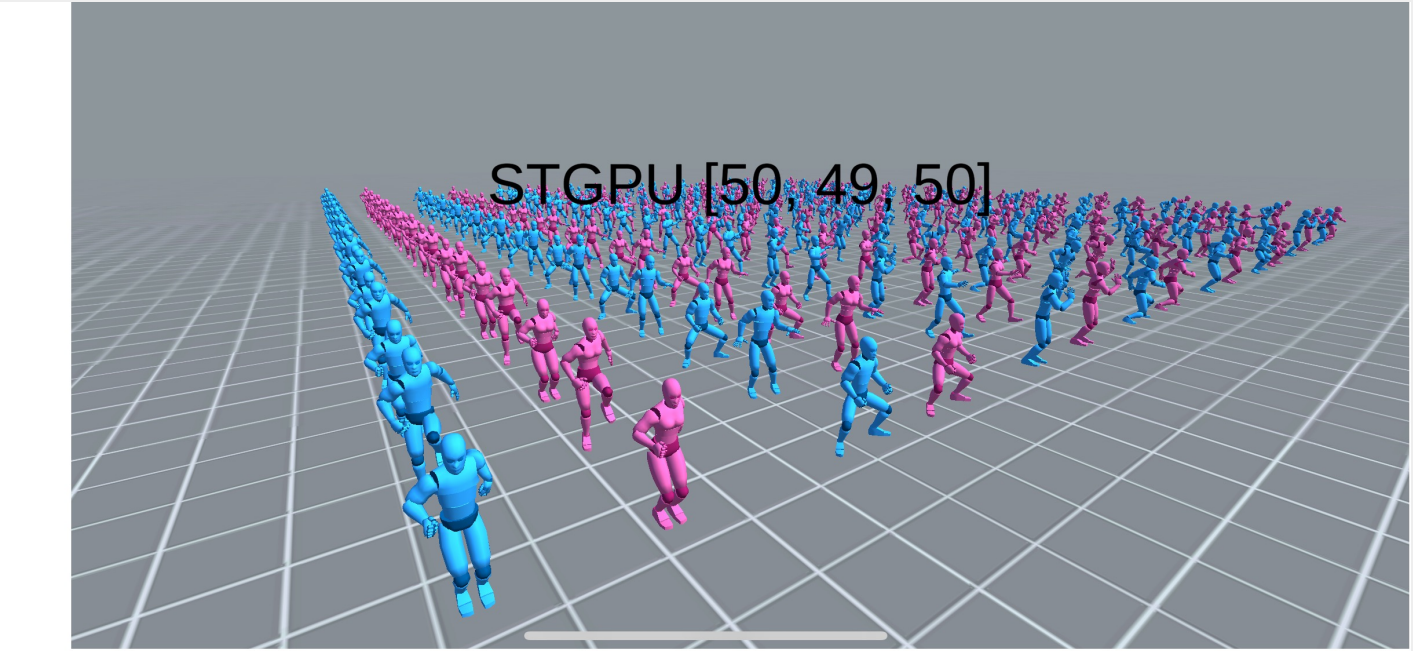


MtSimd

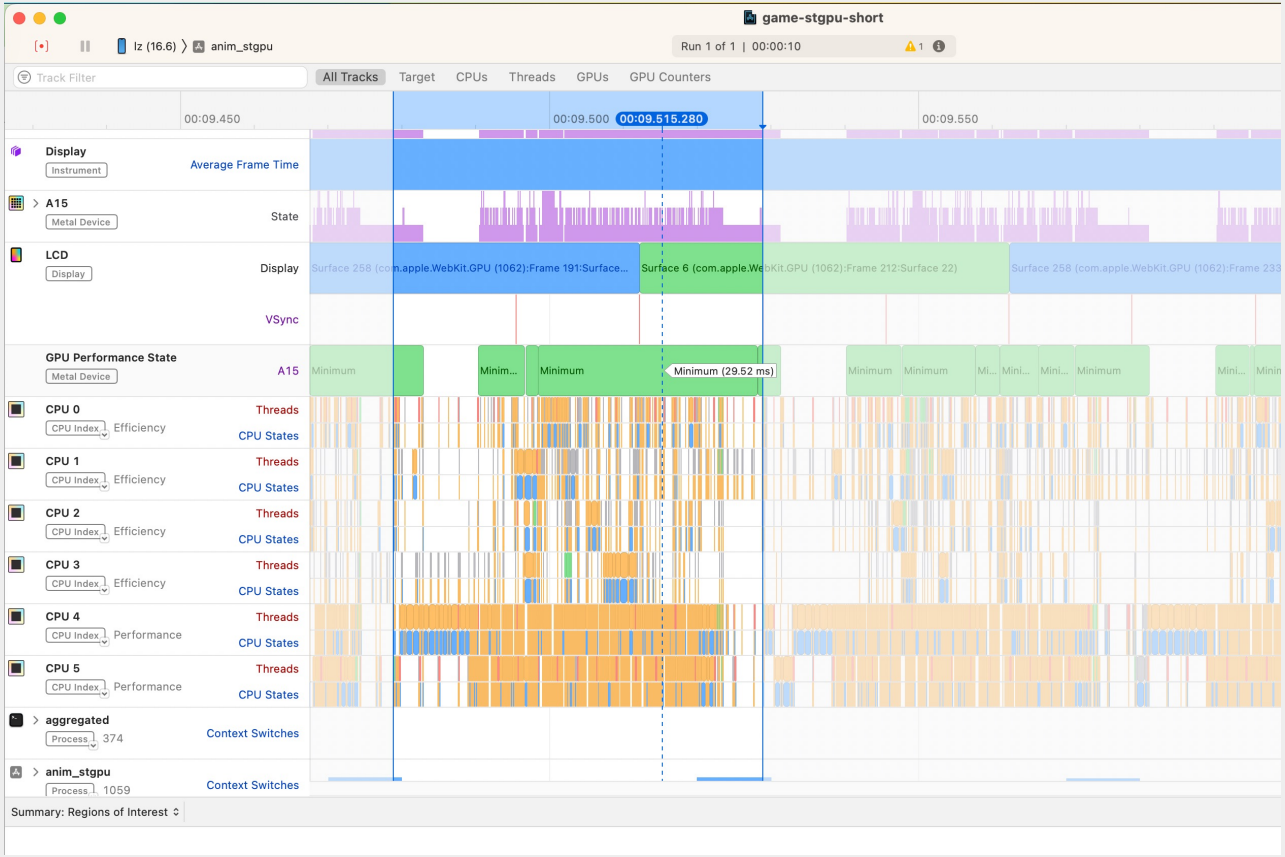


Process...	Process Name	Responsible Process	User Name	% CPU	CPU Time	# Threads	Memory	Kind	Sudden...	Sandbox	R
757	com.apple.WebKit.WebC...	launchd (1)	mobile	151.3%	36.79 s	14	846.57 MiB	arm64	Yes	Yes	N
756	com.apple.WebKit.GPU (...	launchd (1)	mobile	43.3%	6.28 s	13	60.31 MiB	arm64	Yes	Yes	N
67	backboardd (67)	launchd (1)	mobile	12.2%	1.38 min	18	40.24 MiB	arm64	No	Yes	N
0	Unknown (0)	Unknown (0)	root	11.4%	2.26 min	496	174.39 MiB	arm64	No	No	N
474	sysmond (474)	launchd (1)	root	9.1%	37.11 s	4	3.38 MiB	arm64	Yes	No	N
35	mediaserverd (35)	launchd (1)	mobile	7.1%	1.00 min	24	50.64 MiB	arm64	No	Yes	N
451	DTServiceHub (451)	launchd (1)	root	5.4%	1.05 min	9	17.53 MiB	arm64	No	Yes	N
753	anim_mtsimd (753)	launchd (1)	mobile	2.2%	448.32 ms	5	6.50 MiB	arm64	No	Yes	N
287	remotepairingdeviced (2...	launchd (1)	mobile	1.3%	35.07 s	5	5.41 MiB	arm64	No	Yes	N
111	chronod (111)	launchd (1)	mobile	0.8%	13.42 s	3	15.52 MiB	arm64	Yes	Yes	N
137	appleh13camerad (137)	launchd (1)	mobile	0.7%	12.87 s	10	21.64 MiB	arm64	No	Yes	N
475	diagnosticd (475)	launchd (1)	root	0.5%	2.99 s	4	1.67 MiB	arm64	No	No	N
108	com.apple.DriverKit-Appl...	launchd (1)	_driverkit	0.4%	8.11 s	7	21.95 MiB	arm64	No	Yes	N
74	locationd (74)	launchd (1)	root	0.3%	24.79 s	10	17.99 MiB	arm64	No	Yes	N

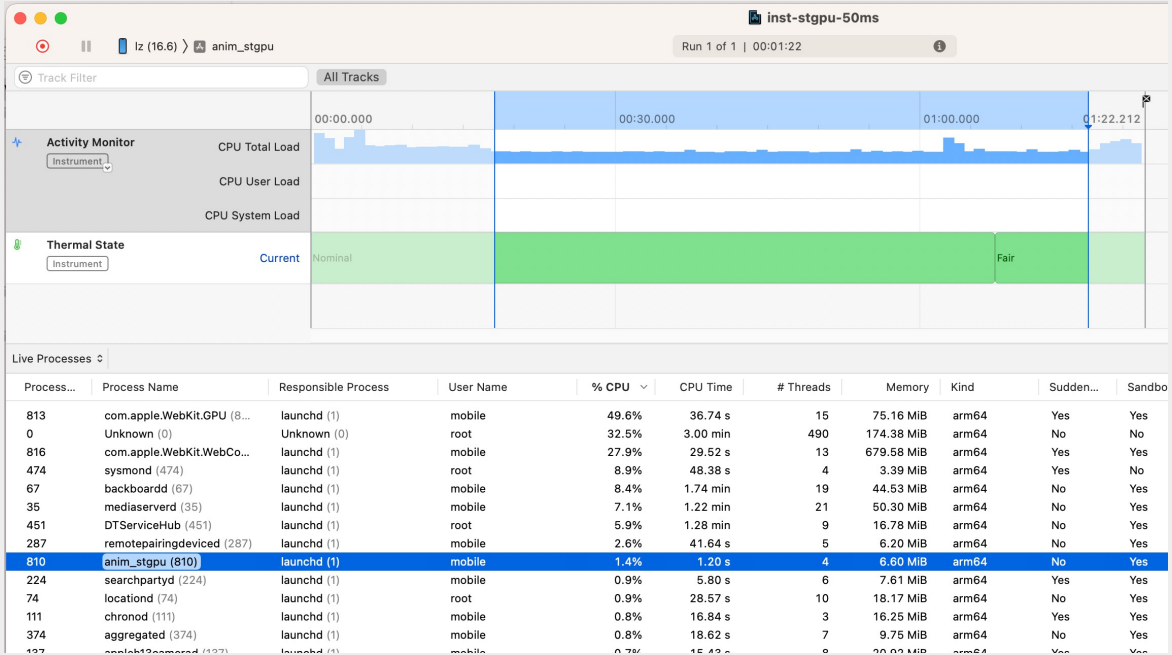
StGpu



StGpu

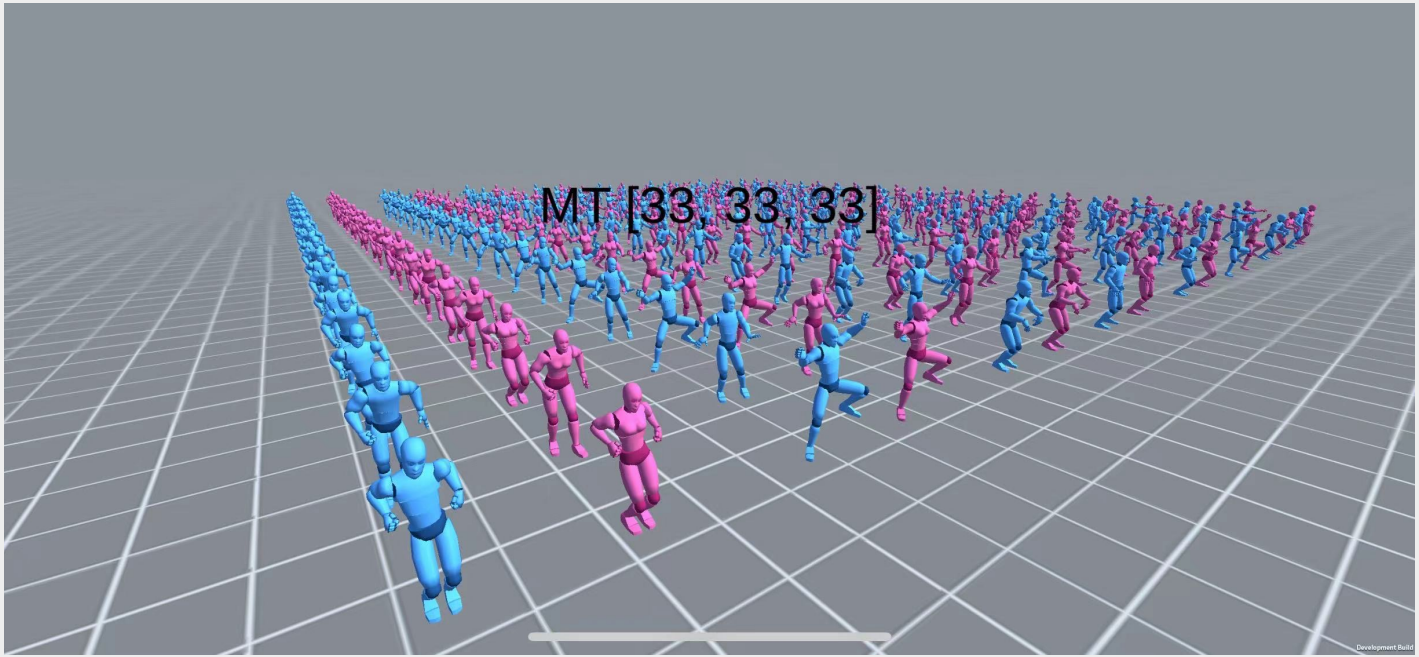


StGpu

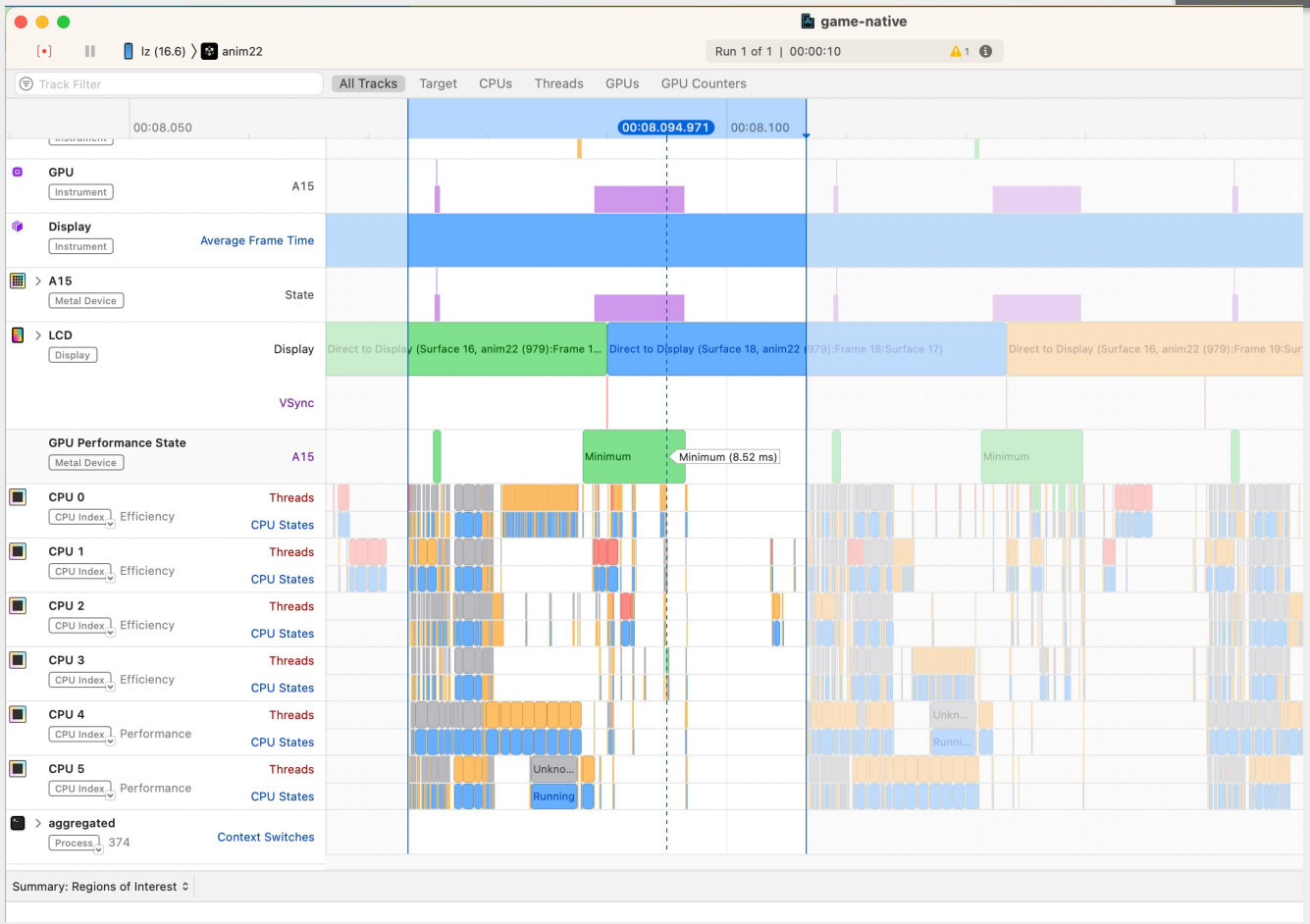


Process...	Process Name	Responsible Process	User Name	% CPU	CPU Time	# Threads	Memory	Kind	Sudden...	Sandbo
813	com.apple.WebKit.GPU (8...	launchd (1)	mobile	49.6%	36.74 s	15	75.16 MiB	arm64	Yes	Yes
0	Unknown (0)	Unknown (0)	root	32.5%	3.00 min	490	174.38 MiB	arm64	No	No
816	com.apple.WebKit.WebCo...	launchd (1)	mobile	27.9%	29.52 s	13	679.58 MiB	arm64	Yes	Yes
474	sysmond (474)	launchd (1)	root	8.9%	48.38 s	4	3.39 MiB	arm64	Yes	No
67	backboardd (67)	launchd (1)	mobile	8.4%	1.74 min	19	44.53 MiB	arm64	No	Yes
35	mediaserverd (35)	launchd (1)	mobile	7.1%	1.22 min	21	50.30 MiB	arm64	No	Yes
451	DTServiceHub (451)	launchd (1)	root	5.9%	1.28 min	9	16.78 MiB	arm64	No	Yes
287	remotepairingdeviced (287)	launchd (1)	mobile	2.6%	41.64 s	5	6.20 MiB	arm64	No	Yes
810	anim_stgpu (810)	launchd (1)	mobile	1.4%	1.20 s	4	6.60 MiB	arm64	No	Yes
224	searchpartyd (224)	launchd (1)	mobile	0.9%	5.80 s	6	7.61 MiB	arm64	Yes	Yes
74	locationd (74)	launchd (1)	root	0.9%	28.57 s	10	18.17 MiB	arm64	No	Yes
111	chronod (111)	launchd (1)	mobile	0.8%	16.84 s	3	16.25 MiB	arm64	Yes	Yes
374	aggregated (374)	launchd (1)	mobile	0.8%	18.62 s	7	9.75 MiB	arm64	No	Yes

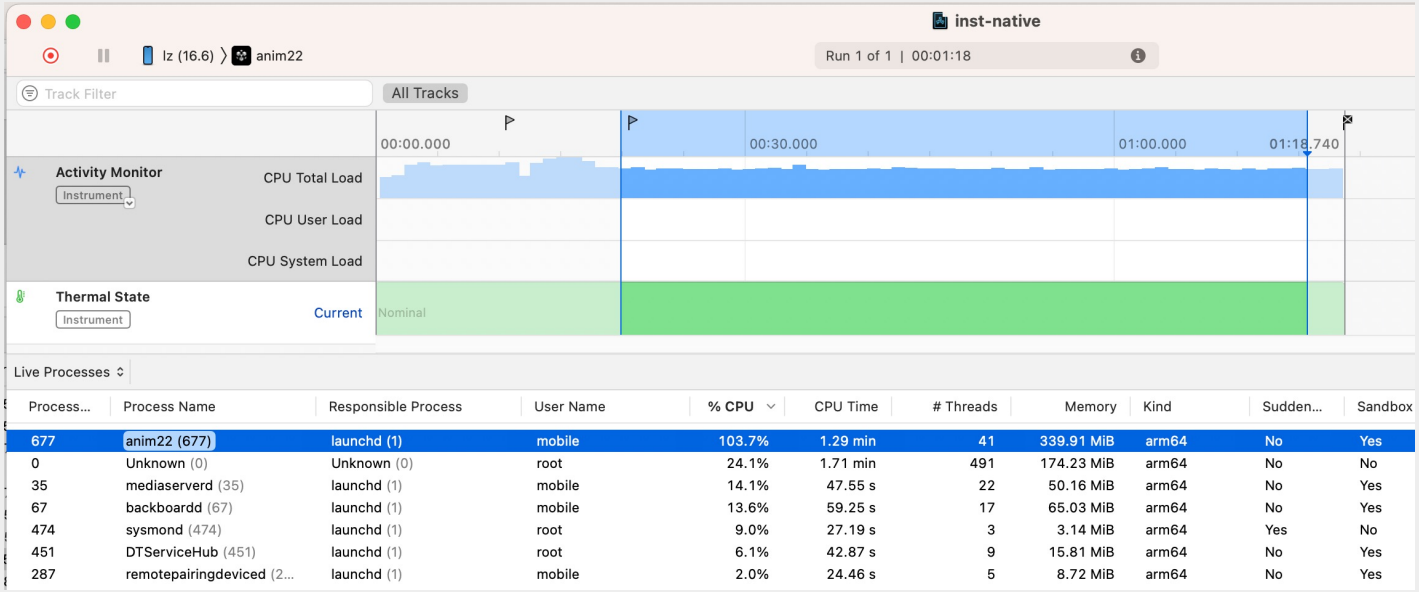
Native



Native



Native



Process...	Process Name	Responsible Process	User Name	% CPU	CPU Time	# Threads	Memory	Kind	Sudden...	Sandbox
677	anim22 (677)	launchd (1)	mobile	103.7%	1.29 min	41	339.91 MiB	arm64	No	Yes
0	Unknown (0)	Unknown (0)	root	24.1%	1.71 min	491	174.23 MiB	arm64	No	No
35	mediaserverd (35)	launchd (1)	mobile	14.1%	47.55 s	22	50.16 MiB	arm64	No	Yes
67	backboardd (67)	launchd (1)	mobile	13.6%	59.25 s	17	65.03 MiB	arm64	No	Yes
474	sysmond (474)	launchd (1)	root	9.0%	27.19 s	3	3.14 MiB	arm64	Yes	No
451	DTServiceHub (451)	launchd (1)	root	6.1%	42.87 s	9	15.81 MiB	arm64	No	Yes
287	remotepairingdeviced (2...	launchd (1)	mobile	2.0%	24.46 s	5	8.72 MiB	arm64	No	Yes

SIMD

Item	Frame time	CPU %	CPU %	Mem (MB)	Mem (MB)	GPU (ms)	GPU
St	66	93.1% 20.9% 1.1%	115.1%	634.97 (WebContent) 59.55 (GPU) 6.49 (宿主App)	701.01	11.57	9.37+2.2 Minimum
StSimd (推荐1)	33 (不稳)	93.4% 39.0% 1.8%	134.2%	686.25 60.31 6.52	753.08	12.17	9.47+2.70 Minimum
MtSimd	33	151.3% 43.3% 2.2%	196.8%	846.57 60.31 6.50	913.38	8.54	8.54 Minimum
StGpu (推荐2)	50	27.9% 49.6% 1.4%	78.9%	679.58 75.16 6.60	761.34	40.51	6.17+1.63+29.52+3.19 Minimum
Native	33	103.7%	103.7%	339.91	339.91	9.13	8.51+0.62 Minimum

Shader Warmup

- → 问题
  - Warmup很慢
  - Block主线程
  - 不支持glProgramBinary
- → 方案
  - 异步warmup
  - 宿主添加 glProgramBinary支持

		WebGL2	Native (Android: Gles3 iOS: Metal)
小米8 (Chrome)	第一次	1138.5 ms	958.6 ms
	第二次	1015.8 ms	34 ms
小米11 Ultra (Chrome)	第一次	337 ms	716 ms
	第二次	331 ms	38.43 ms
iPhone SE3 (Safari, via Metal on)	第一次	748 ms	1041.85 ms
	第二次	796 ms	22.57 ms
iPhone SE3 (Safari, via Metal off)	第一次	149 ms	1041.85 ms
	第二次	149 ms	22.57 ms

Wasm+native

- → wasm代码减少，大大减少内存
- → 充分使用Native能力
  - Vulkan，Metal
  - Native高效的计算能力
  - 文件访问能力

未来工作

多线程

- 问题
  - 内存占用大?
  - CPU总开销大?
  - 当前不支持RenderThread，正在解决
  - Web worker不能访问DOM
  - 只能支持native代码，不支持c#代码
- 优点
  - 利用多核心，提高帧率
  - 同样的计算量，功耗更小

WebGPU

→ 支持更多功能

- Compute Pipeline
  - GPU skinning
  - GPU particles
  - GPU culling
  - Post Processing
- Indirect draw
- Debug easily
  - Label
  - Debug group
- …

References:
https://www.khronos.org/assets/uploads/developers/presentations/WebGPU_Best_Practices_Google.pdf

WebGPU

→ 更高效率

- CommandBuffer + Queue
- RenderBundle(预录制，可复用)
- RenderPipeline(支持异步创建)
- ShaderModule(允许多程序入口)
- 更好地支持多线程

References:
https://gpuweb.github.io/gpuweb/explainer
https://github.com/gpuweb/gpuweb/wiki/The-Multi-Explainer

Thank you!

Let’s keep in touch - liang.zhao@unity.cn